Foundations of Probabilistic Answers to Queries

Dan Suciu and Nilesh Dalvi
University of Washington
Databases Today are Deterministic

• An item either is in the database or is not

• A tuple either is in the query answer or is not

• This applies to all variety of data models:
  – Relational, E/R, NF2, hierarchical, XML, …
What is a Probabilistic Database?

• “An item belongs to the database” is a probabilistic event

• “A tuple is an answer to the query” is a probabilistic event

• Can be extended to all data models; we discuss only probabilistic relational data
Two Types of Probabilistic Data

• Database is deterministic
  Query answers are probabilistic

• Database is probabilistic
  Query answers are probabilistic
Long History

Probabilistic relational databases have been studied from the late 80’s until today:

- Cavallo & Pitarelli: 1987
- Barbara, Garcia-Molina, Porter: 1992
- Lakshmanan, Leone, Ross & Subrahmanian: 1997
- Fuhr & Roellke: 1997
- Dalvi & S: 2004
- Widom: 2005
So, Why Now?

Application pull:
• The need to manage imprecisions in data

Technology push:
• Advances in query processing techniques

The tutorial is built on these two themes
Application Pull

Need to manage imprecisions in data

- Many types: non-matching data values, imprecise queries, inconsistent data, misaligned schemas, etc, etc

The quest to manage imprecisions = major driving force in the database community

- Ultimate cause for many research areas: data mining, semistructured data, schema matching, nearest neighbor
Theme 1:

A *large* class of imprecisions in data can be modeled with probabilities
Technology Push

Processing probabilistic data is fundamentally more complex than other data models

• Some previous approaches sidestepped complexity

There exists a rich collection of powerful, non-trivial techniques and results, some old, some very recent, that could lead to practical management techniques for probabilistic databases.
Theme 2:

Identify the source of complexity, present snapshots of non-trivial results, set an agenda for future research.
Some Notes on the Tutorial

There is a huge amount of related work:
probabilistic db, top-k answers, KR, probabilistic reasoning, random graphs, etc, etc.

We left out many references
All references used are available in separate document


Requires TexPoint to view http://www.thp.uni-koeln.de/~ang/texpoint/index.html
Overview

Part I: Applications: Managing Imprecisions
Part II: A Probabilistic Data Semantics
Part III: Representation Formalisms
Part IV: Theoretical foundations
Part V: Algorithms, Implementation Techniques
Summary, Challenges, Conclusions
Part I

Applications: Managing Imprecisions
Outline

1. Ranking query answers
2. Record linkage
3. Quality in data integration
4. Inconsistent data
5. Information disclosure
1. Ranking Query Answers

Database is deterministic

The query returns a *ranked list of tuples*

- User interested in top-k answers.
The Empty Answers Problem

Query is overspecified: no answers
Example: try to buy a house in

```
SELECT *
FROM Houses
WHERE bedrooms = 4
     AND  style = 'craftsman'
     AND  district = 'View Ridge'
     AND  price < 400000
```

… good luck!

Today users give up and move to Baltimore
Ranking:
Compute a similarity score between a tuple and the query \( Q \):

\[
Q = \text{SELECT } * \quad \text{FROM} \quad R \quad \text{WHERE} \quad A_1=v_1 \quad \text{AND} \ldots \quad \text{AND} \quad A_m=v_m
\]

Query is a vector:
\[
Q = (v_1, \ldots, v_m)
\]

Tuple is a vector:
\[
T = (u_1, \ldots, u_m)
\]

Rank tuples by their TF/IDF similarity to the query \( Q \)

Includes partial matches
Similarity Predicates in SQL

Beyond a single table:
“Find the good deals in a neighborhood!”

```
SELECT *
FROM Houses x
WHERE x.bedrooms ~ 4 AND x.style ~ 'craftsman' AND x.price ~ 600k
     AND NOT EXISTS 
     (SELECT *
      FROM Houses y
      WHERE x.district = y.district AND x.ID != y.ID
           AND y.bedrooms ~ 4 AND y.style ~ 'craftsman' AND y.price ~ 600k
     )
```

Users specify similarity predicates with ~
System combines atomic similarities using probabilities
Types of Similarity Predicates

• String edit distances:
  – Levenstein distance, Q-gram distances

• TF/IDF scores

• Ontology distance / semantic similarity:
  – Wordnet

• Phonetic similarity:
  – SOUNDEX

Keyword Searches in Databases

Goal:
• Users want to search via keywords
• Do not know the schema

Techniques:
• Matching objects may be scattered across physical tables due to normalization; need on the fly joins
• Score of a tuple = number of joins, plus “prestige” based on indegree

[Hristidis&Papakonstantinou’2002,Bhalotia et al.2002]
Q = ‘Abiteboul’ and ‘Widom’
More Ranking: User Preferences

Applications: personalized search engines, shopping agents, logical user profiles, “soft catalogs”

Two approaches:

• Qualitative $\Rightarrow$ Pareto semantics (deterministic)
• Quantitative $\Rightarrow$ alter the query ranking

Summary on Ranking Query Answers

Types of imprecision addressed:
Data is precise, query answers are imprecise:
• User has limited understanding of the data
• User has limited understanding of the schema
• User has personal preferences

Probabilistic approach would…
• Principled semantics for complex queries
• Integrate well with other types of imprecision
2. Record Linkage

Determine if two data records describe same object

Scenarios:

• Join/merge two relations
• Remove duplicates from a single relation
• Validate incoming tuples against a reference
Fellegi-Sunter Model

A probabilistic model/framework

• Given two sets of records A, B:

Goal: partition $A \times B$ into:

• Match
• Uncertain
• Non-match

$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

$B = \{b_1, b_2, b_3, b_4, b_5\}$
Non-Fellegi Sunter Approaches

**Deterministic** linkage

- Normalize records, then test equality
  - E.g. for addresses
  - Very fast when it works
- Hand-coded rules for an “acceptable match”
  - E.g. “same SSN”; or “same last name AND same DOB”
  - Difficult to tune
Application: Data Cleaning, ETL

• Merge/purge for large databases, by sorting and clustering
  [Hernandez, Stolfo: 1995]

• Use of dimensional hierarchies in data warehouses and exploit co-occurrences
  [Ananthakrishna, Chaudhuri, Ganti: 2002]

• Novel similarity functions that are amenable to indexing
  [Chaudhuri, Ganjam, Ganti, Motwani: 2002]

• Declarative language to combine cleaning tasks
  [Galhardas et al.: 2001]
Application: Data Integration

WHIRL

- All attributes in all tables are of type text
- Datalog queries with two kinds of predicates:
  - Relational predicates
  - Similarity predicates $X \sim Y$

Matches two sets on the fly, but not really a “record linkage” application.

[Cohen:1998]
WHIRL

Example 1:

\[ Q_1(*) \ :- \ P(\text{Company}_1, \text{Industry}_1), \]
\[ Q(\text{Company}_2, \text{Website}), \]
\[ R(\text{Industry}_2, \text{Analysis}), \]
\[ \text{Company}_1 \sim \text{Company}_2, \]
\[ \text{Industry}_1 \sim \text{Industry}_2 \]

Score of an answer tuple = product of similarities
WHIRL

Example 2 (with projection):

\[ Q_2(\text{Website}) :- P(\text{Company}_1, \text{Industry}_1), \]
\[ Q(\text{Company}_2, \text{Website}), \]
\[ R(\text{Industry}_2, \text{Analysis}), \]
\[ \text{Company}_1 \sim \text{Company}_2, \]
\[ \text{Industry}_1 \sim \text{Industry}_2 \]

\[ \text{Support}(t) = \text{set of tuples supporting the answer } t \]

\[ \text{score}(t) = 1 - \prod_{s \in \text{Support}(t)} (1 - \text{score}(s)) \]
Summary on Record Linkage

Types of imprecision addressed:
Same entity represented in different ways
• Misspellings, lack of canonical representation, etc.

A probability model would…
• Allow system to use the match probabilities: cheaper, on-the-fly
• But need to model complex probabilistic correlations: is one set a reference set? how many duplicates are expected?
3. Quality in Data Integration

Use of probabilistic information to reason about soundness, completeness, and overlap of sources

Applications:

- Order access to information sources
- Compute confidence scores for the answers

[Florescu,Koller,Levy97;Chang,GarciaMolina00;Mendelzon,Mihaila01]
Global Historical Climatology Network

- Integrates climatic data from:
  - 6000 temperature stations
  - 7500 precipitation stations
  - 2000 pressure stations

**Soundness** of a data source:
what fraction of items are correct

**Completeness** data source:
what fractions of items it actually contains
Local as

\[
S_1: \\
V_1(s, \text{lat}, \text{lon}, c) \rightarrow \text{Station}(s, \text{lat}, \text{lon} c)
\]

\[
S_2: \\
V_2(s, y, m, v) \rightarrow \\
\text{Temperature}(s, y, m, v), \\
\text{Station}(s, \text{lat}, \text{lon}, \text{"Canada"}), y \geq 1900
\]

\[
S_3: \\
V_3(s, y, m, v) \rightarrow \\
\text{Temperature}(s, y, m, v), \\
\text{Station}(s, \text{lat}, \text{lon}, \text{"US"}), y \geq 1800
\]

\[
S_{8756}: \\
\ldots
\]

Global schema: \text{Temperature Station}
Next, declare soundness and completeness:

\[
S_2: \quad V_2(s, y, m, v) \rightarrow
\]

\textbf{Temperature}(s, y, m, v), \quad \textbf{Station}(s, \text{lat}, \text{lon}, \text{"Canada"}), \quad y \geq 1900

\text{Soundness}(V_2) \geq 0.7
\text{Completeness}(V_2) \geq 0.4
Goal 1: completeness → order source accesses

\[ S_5 \quad S_{74} \quad S_2 \quad S_{31} \quad \ldots \]

Goal 2: soundness → query confidence

\[
\text{Q}(y, v) :\quad \text{Temperature}(s, y, m, v), \quad \text{Station}(s, \text{lat}, \text{lon}, \text{“US”}), \\
y \geq 1950, \; y \leq 1955, \; \text{lat} \geq 48, \; \text{lat} \leq 49
\]

Answer:

<table>
<thead>
<tr>
<th>Year</th>
<th>Value</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1952</td>
<td>55° F</td>
<td>0.7</td>
</tr>
<tr>
<td>1954</td>
<td>-22° F</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

---

[Florescu,Koller,Levy:1997]

[Mendelzon&Mihaila:2001]
Summary: Quality in Data Integration

Types of imprecision addressed
Overlapping, inconsistent, incomplete data sources

- Data is probabilistic
- Query answers are probabilistic

They use already a probabilistic model

- Needed: complex probabilistic spaces. E.g. a tuple \( t \) in \( V_1 \) has 60% probability of also being in \( V_2 \)
- Query processing still in infancy
4. Inconsistent Data

Goal:

consistent query answers
from inconsistent databases

Applications:

• Integration of autonomous data sources
• Un-enforced integrity constraints
• Temporary inconsistencies

[Bertosi&Chomicki:2003]
The Repair Semantics

Consider all “repairs”

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>State</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miklau</td>
<td>UW</td>
<td>WA</td>
<td>Data security</td>
</tr>
<tr>
<td>Dalvi</td>
<td>UW</td>
<td>WA</td>
<td>Prob. Data</td>
</tr>
<tr>
<td>Balazinska</td>
<td>UW</td>
<td>WA</td>
<td>Data streams</td>
</tr>
<tr>
<td>Balazinska</td>
<td>MIT</td>
<td>MA</td>
<td>Data streams</td>
</tr>
<tr>
<td>Miklau</td>
<td>Umass</td>
<td>MA</td>
<td>Data security</td>
</tr>
</tbody>
</table>

Find people in State=W A  ⇒ Dalvi

Find people in State=MA  ⇒ ∅

Hi precision, but low recall
Alternative Probabilistic Semantics

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>State</th>
<th>Area</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miklau</td>
<td>UW</td>
<td>WA</td>
<td>Data security</td>
<td>0.5</td>
</tr>
<tr>
<td>Dalvi</td>
<td>UW</td>
<td>WA</td>
<td>Prob. Data</td>
<td>1</td>
</tr>
<tr>
<td>Balazinska</td>
<td>UW</td>
<td>WA</td>
<td>Data streams</td>
<td>0.5</td>
</tr>
<tr>
<td>Balazinska</td>
<td>MIT</td>
<td>MA</td>
<td>Data streams</td>
<td>0.5</td>
</tr>
<tr>
<td>Miklau</td>
<td>Umass</td>
<td>MA</td>
<td>Data security</td>
<td>0.5</td>
</tr>
</tbody>
</table>

State=WA  ⇒  Dalvi, Balazinska(0.5), Miklau(0.5)

State=MA  ⇒  Balazinska(0.5), Miklau(0.5)

**Lower precision, but better recall**
Summary: Inconsistent Data

Types of imprecision addressed:
• Data from different sources is contradictory
• Data is uncertain, hence, arguably, probabilistic
• Query answers are probabilistic

A probabilistic would…
• Give better recall!
• Needs to support disjoint tuple events
5. Information Disclosure

Goal

- Disclose some information (V) while protecting private or sensitive data S

Applications:

- Privacy preserving data mining
- Data exchange
- K-anonymous data

V = anonymized transactions
V = standard view(s)
V = k-anonymous table

S = some atomic fact that is private
Pr(S) = a prior probability of S

Pr(S | V) = a posterior probability of S
Information Disclosure

- If $\rho_1 < \rho_2$, a $\rho_1$, $\rho_2$ privacy breach:
  \[ \Pr(S) \leq \rho_1 \quad \text{and} \quad \Pr(S \mid V) \geq \rho_2 \]

- Perfect security:
  \[ \Pr(S) = \Pr(S \mid V) \]

- Practical security:
  \[ \lim_{\text{domain size} \to \infty} \Pr(S \mid V) = 0 \]
Summary: Information Disclosure

Is this a type of imprecision in data?
• Yes: it’s the adversary’s uncertainty about the private data.
• The only type of imprecision that is good

Techniques
• Probabilistic methods: long history [Shannon’49]
• Definitely need conditional probabilities
Summary: Information Disclosure

Important fundamental duality:

- Query answering: want Probability $\leq 1$
- Information disclosure: want Probability $\geq 0$

They share the same fundamental concepts and techniques
Summary: Information Disclosure

What is required from the probabilistic model

• Don’t know the possible instances
• Express the adversary’s knowledge:
  – Cardinalities:
  – Correlations between values:
• Compute conditional probabilities

Size(Employee) ~ 1000
area-code ~→ city
6. Other Applications

- Data lineage + accuracy: Trio [Widom:2005]
- Sensor data [Deshpande, Guestrin, Madden:2004]
- Personal information management
  Semex [Dong & Halevy:2005, Dong, Halevy, Madhavan:2005]
  Heystack [Karger et al. 2003], Magnet [Sinha & Karger:2005]
- Using statistics to answer queries [Dalvi & S; 2005]
Summary on Part I: Applications

Common in these applications:

- Data in database and/or in query answer is uncertain, ranked; sometimes probabilistic

Need for common probabilistic model:

- Main benefit: uniform approach to imprecision
- Other benefits:
  - Handle complex queries (instead of single table TF/IDF)
  - Cheaper solutions (on-the-fly record linkage)
  - Better recall (constraint violations)
Part II

A Probabilistic Data Semantics
Outline

• The possible worlds model

• Query semantics
Possible Worlds Semantics

Attribute domains: \(\text{int, char}(30), \text{varchar}(55), \text{datetime}\)

\# values: \(2^{32}, 2^{120}, 2^{440}, 2^{64}\)

Relational schema:

\(\text{Employee}(\text{name:varchar}(55), \text{dob:datetime}, \text{salary:int})\)

\# of tuples: \(2^{440} \times 2^{64} \times 2^{23}\)

Database schema:

\(\text{Employee}(\ldots), \text{Projects}(\ldots), \text{Groups}(\ldots), \text{WorksFor}(\ldots)\)

\# of instances: \(N (= \text{BIG but finite})\)
The Definition

The set of all possible database instances:

\[ \text{INST} = \{I_1, I_2, I_3, \ldots, I_N\} \]

**Definition** A *probabilistic database* \( I^p \) is a probability distribution on INST

\[ \text{Pr} : \text{INST} \rightarrow [0,1] \quad \text{s.t.} \quad \sum_{i=1}^{N} \text{Pr}(I_i) = 1 \]

**Definition** A *possible world* is I s.t. \( \text{Pr}(I) > 0 \)
\[ I_p = \text{Example} \]

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

\[ \Pr(I_1) = 1/3 \]

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

\[ \Pr(I_3) = 1/2 \]

Possible worlds = \{I_1, I_2, I_3, I_4\}

\[ \Pr(I_2) = 1/12 \]

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

\[ \Pr(I_4) = 1/12 \]
Tuples as Events

One tuple $t$  $\Rightarrow$ event $t \in I$

$$Pr(t) = \sum_{I: t \in I} Pr(I)$$

Two tuples $t_1, t_2$  $\Rightarrow$ event $t_1 \in I \land t_2 \in I$

$$Pr(t_1 \land t_2) = \sum_{I: t_1 \in I \land t_2 \in I} Pr(I)$$
Tuple Correlation

- **Disjoint**
  \[ \Pr(t_1 t_2) = 0 \]

- **Negatively correlated**
  \[ \Pr(t_1 t_2) < \Pr(t_1) \Pr(t_2) \]

- **Independent**
  \[ \Pr(t_1 t_2) = \Pr(t_1) \Pr(t_2) \]

- **Positively correlated**
  \[ \Pr(t_1 t_2) > \Pr(t_1) \Pr(t_2) \]

- **Identical**
  \[ \Pr(t_1 t_2) = \Pr(t_1) = \Pr(t_2) \]
**Example**

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

Pr(I₁) = 1/3

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Pr(I₃) = 1/2

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

Pr(I₂) = 1/12

<table>
<thead>
<tr>
<th>Customer</th>
<th>Address</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Pr(I₄) = 1/12
Query Semantics

Given a query $Q$ and a probabilistic database $I^p$, what is the meaning of $Q(I^p)$?
Query Semantics

Semantics 1: Possible Answers
A probability distribution on \textit{sets of tuples}

\[
\forall A. \Pr(Q = A) = \sum_{I \in \text{INST. } Q(I) = A} \Pr(I)
\]

Semantics 2: Possible Tuples
A probability function on \textit{tuples}

\[
\forall t. \Pr(t \in Q) = \sum_{I \in \text{INST. } t \in Q(I)} \Pr(I)
\]
**Example: Query Semantics**

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Camera</td>
</tr>
</tbody>
</table>

\[ \text{Pr}(I_1) = 1/3 \]

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
</tbody>
</table>

\[ \text{Pr}(I_2) = 1/12 \]

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Gadget</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

\[ \text{Pr}(I_3) = 1/2 \]

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

\[ \text{Pr}(I_4) = 1/12 \]

**Possible answers semantics:**

<table>
<thead>
<tr>
<th>Answer set</th>
<th>Probability</th>
<th>\text{Pr}(I_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo, Camera</td>
<td>1/3</td>
<td>\text{Pr}(I_1)</td>
</tr>
<tr>
<td>Gizmo</td>
<td>1/12</td>
<td>\text{Pr}(I_2)</td>
</tr>
<tr>
<td>Camera</td>
<td>7/12</td>
<td>\text{P}(I_3) + \text{P}(I_4)</td>
</tr>
</tbody>
</table>

**Possible tuples semantics:**

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
<th>\text{Pr}(I_i) + \text{P}(I_3) + \text{P}(I_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>11/12</td>
<td>\text{Pr}(I_1) + \text{P}(I_2)</td>
</tr>
<tr>
<td>Gizmo</td>
<td>5/12</td>
<td>\text{Pr}(I_1) + \text{Pr}(I_2)</td>
</tr>
</tbody>
</table>
**Special Case**

**Tuple independent probabilistic database**

\[ \text{Pr}(I) = \prod_{t \in I} \text{pr}(t) \times \prod_{t \notin I} (1-\text{pr}(t)) \]

\( \text{pr} : \text{TUP} \rightarrow [0,1] \)

\( \text{TUP} = \{t_1, t_2, \ldots, t_M\} = \text{all tuples} \)

\( \text{INST} = \mathcal{P}(\text{TUP}) \)

\( N = 2^M \)

\( \text{No restrictions} \)
## Tuple Prob. ⇒ Possible Worlds

### Joint Probabilities

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>0.8</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>0.6</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>0.9</td>
</tr>
</tbody>
</table>

### Expected Size of \( I^p \)

\[ E[ \text{size}(I^p) ] = \frac{2.3}{62} \text{ tuples} = 0.037 \]

### Joint Probabilities

\[
\sum_{I^p} = 1
\]
**Tuple Prob. ⇒ Query Evaluation**

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>$p_1$</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>$p_2$</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>

**Customer** | **Product** | **Date** | **pr**
---|---|---|---
John | Gizmo | . . . | $q_1$
John | Gadget | . . . | $q_2$
John | Gadget | . . . | $q_3$
Sue | Camera | . . . | $q_4$
Sue | Gadget | . . . | $q_5$
Sue | Gadget | . . . | $q_6$
Fred | Gadget | . . . | $q_7$

**SELECT DISTINCT** $x$.city 
**FROM** Person $x$, Purchase $y$ 
**WHERE** $x$.Name = $y$.Customer and $y$.Product = ‘Gadget’

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>$p_1(1-(1-q_2)(1-q_3))$</td>
</tr>
<tr>
<td>Boston</td>
<td>$1 - (1 - p_2(1-(1-q_5)(1-q_6))) \times (1 - p_3 q_7 )$</td>
</tr>
</tbody>
</table>
Summary of Part II

Possible Worlds Semantics

• Very powerful model: any tuple correlations
• Needs separate representation formalism
Summary of Part II

Query semantics

- Very powerful: *every* SQL query has semantics
- Very intuitive: from standard semantics
- Two variations, both appear in the literature
Summary of Part II

Possible answers semantics

• Precise
• Can be used to compose queries
• Difficult user interface

Possible tuples semantics

• Less precise, but simple; sufficient for most apps
• Cannot be used to compose queries
• Simple user interface
After the Break

Part III: Representation Formalisms

Part IV: Foundations

Part V: Algorithms, implementation techniques

Conclusions and Challenges
Part III

Representation Formalisms
Representation Formalisms

Problem
Need a good representation formalism

- Will be interpreted as possible worlds
- Several formalisms exist, but no winner
Evaluation of Formalisms

• What possible worlds can it represent?
• What probability distributions on worlds?
• Is it closed under query application?
Outline

A complete formalism:
• Intensional Databases

Incomplete formalisms:
• Various expressibility/complexity tradeoffs
Intensional Database

Atomic event ids

$e_1, e_2, e_3, \ldots$

Probabilities:

$p_1, p_2, p_3, \ldots \in [0,1]$

Event expressions: $\land, \lor, \neg$

$e_3 \land (e_5 \lor \neg e_2)$

Intensional probabilistic database $J$:

each tuple $t$ has an event attribute $t.E$
### Intensional DB ⇒ Possible Worlds

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>$e_1 \land (e_2 \lor e_3)$</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>$(e_1 \land e_2) \lor (e_2 \land e_3)$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
J &= \{ 000, 001, 010, 011, 100, 101, 110, 111 \} \\
\mathbf{p} &= (1-p_1)(1-p_2)(1-p_3) + (1-p_1)(1-p_2)p_3 + (1-p_1)p_2(1-p_3) + p_1(1-p_2)(1-p_3) + p_1p_2(1-p_3) + p_1p_2p_3 \\
e_1e_2e_3 &= 000, 001, 010, 011, 100, 101, 110, 111
\end{align*}
\]
Possible Worlds $\Rightarrow$ Intensional DB

$$
\begin{align*}
E_1 &= e_1 \\
E_2 &= \neg e_1 \land e_2 \\
E_3 &= \neg e_1 \land \neg e_2 \land e_3 \\
E_4 &= \neg e_1 \land \neg e_2 \land \neg e_3 \land e_4
\end{align*}
$$

$$
\begin{align*}
\Pr(e_1) &= p_1 \\
\Pr(e_2) &= p_2/(1-p_1) \\
\Pr(e_3) &= p_3/(1-p_1-p_2) \\
\Pr(e_4) &= p_4/(1-p_1-p_2-p_3)
\end{align*}
$$

Intensional DBs are complete
Closure Under Operators

One still needs to compute probability of event expression
Summary on Intensional Databases

Event expression for each tuple
- Possible worlds: any subset
- Probability distribution: any

Complete (in some sense) … but impractical

Important abstraction: consider restrictions

Related to c-tables [Imilelinski&Lipski:1984]
Restricted Formalisms

Explicit tuples

• Have a tuple template for every tuple that may appear in a possible world

Implicit tuples

• Specify tuples indirectly, e.g. by indicating how many there are
Explicit Tuples

Independent tuples

tuple = event

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>E</th>
<th>pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>e₁</td>
<td>0.8</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>e₂</td>
<td>0.2</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>e₃</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Atomic, distinct. May use TIDs.

\[ E[ \text{size(Customer)} ] = 1.6 \text{ tuples} \]
Application 1: Similarity Predicates

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Profession</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>statistician</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>musician</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>physicist</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cust</th>
<th>Product</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Gizmo</td>
<td>dishware</td>
</tr>
<tr>
<td>John</td>
<td>Gadget</td>
<td>instrument</td>
</tr>
<tr>
<td>John</td>
<td>Gadget</td>
<td>instrument</td>
</tr>
<tr>
<td>Sue</td>
<td>Camera</td>
<td>musicware</td>
</tr>
<tr>
<td>Sue</td>
<td>Gadget</td>
<td>microphone</td>
</tr>
<tr>
<td>Sue</td>
<td>Gadget</td>
<td>instrument</td>
</tr>
<tr>
<td>Fred</td>
<td>Gadget</td>
<td>microphone</td>
</tr>
</tbody>
</table>

SELECT DISTINCT x.city
FROM Person x, Purchase y
WHERE x.Name = y.Cust
  and y.Product = 'Gadget'
  and x.profession ~ 'scientist'
  and y.category ~ 'music'
## Application 1: Similarity Predicates

### Table: Name, City, Profession, and Similarity Predicates

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Profession</th>
<th>pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>statistician</td>
<td>$p_1 = 0.8$</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>musician</td>
<td>$p_2 = 0.2$</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>physicist</td>
<td>$p = 0.9$</td>
</tr>
</tbody>
</table>

### Table: Customer (Cust), Product, Category, and Similarity Predicates

<table>
<thead>
<tr>
<th>Cust</th>
<th>Product</th>
<th>Category</th>
<th>pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Gizmo</td>
<td>dishware</td>
<td>$q_1 = 0.2$</td>
</tr>
<tr>
<td>John</td>
<td>Gadget</td>
<td>instrument</td>
<td>$q_2 = 0.6$</td>
</tr>
<tr>
<td>John</td>
<td>Gadget</td>
<td>instrument</td>
<td>$q_3 = 0.6$</td>
</tr>
<tr>
<td>Sue</td>
<td>Camera</td>
<td>musicware</td>
<td>$q_4 = 0.9$</td>
</tr>
<tr>
<td>Sue</td>
<td>Gadget</td>
<td>microphone</td>
<td>$q_5 = 0.7$</td>
</tr>
<tr>
<td>Sue</td>
<td>Gadget</td>
<td>instrument</td>
<td>$q_6 = 0.6$</td>
</tr>
<tr>
<td>Fred</td>
<td>Gadget</td>
<td>microphone</td>
<td>$q_7 = 0.7$</td>
</tr>
</tbody>
</table>

### SQL Query

```sql
SELECT DISTINCT x.city
FROM Person p x, Purchase p y
WHERE x.Name = y.Cust
  AND y.Product = 'Gadget'
  AND x.profession ~ 'scientist'
  AND y.category ~ 'music'
```

### Tuple Probabilities

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>$p_1(1-(1-q_2)(1-q_3))$</td>
</tr>
<tr>
<td>Boston</td>
<td>$1-(1-p_2(1-(1-q_5)(1-q_6)))(1-p_3q_7)$</td>
</tr>
</tbody>
</table>
Explicit Tuples

Independent/disjoint tuples

Independent events: \( e_1, e_2, \ldots, e_i, \ldots \)

Split \( e_i \) into disjoint “shares” \( e_i = e_{i1} \lor e_{i2} \lor e_{i3} \lor \ldots \)

\( e_{34}, e_{37} \Rightarrow \) disjoint events

\( e_{37}, e_{57} \Rightarrow \) independent events
## Application 2: Inconsistent Data

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
</tr>
<tr>
<td>John</td>
<td>Huston</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
</tr>
</tbody>
</table>

### Step 1: resolve violations

Name → City (violated)

```sql
SELECT DISTINCT Product
FROM Customer
WHERE City = 'Seattle'
```
Application 2: Inconsistent Data

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Product</th>
<th>E</th>
<th>Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Gizmo</td>
<td>e_{11}</td>
<td>1/3</td>
</tr>
<tr>
<td>John</td>
<td>Seattle</td>
<td>Camera</td>
<td>e_{11}</td>
<td>1/3</td>
</tr>
<tr>
<td>John</td>
<td>Boston</td>
<td>Gadget</td>
<td>e_{12}</td>
<td>1/3</td>
</tr>
<tr>
<td>John</td>
<td>Huston</td>
<td>Gizmo</td>
<td>e_{13}</td>
<td>1/3</td>
</tr>
<tr>
<td>Sue</td>
<td>Denver</td>
<td>Gizmo</td>
<td>e_{21}</td>
<td>1/2</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>Camera</td>
<td>e_{22}</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Step 1: resolve violations

Step 2: evaluate query

SELECT DISTINCT Product
FROM Customer
WHERE City = 'Seattle'

\[ E[\text{size(Customer)}] = 2 \text{ tuples} \]

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>( p_{11} = 1/3 )</td>
</tr>
<tr>
<td>Camera</td>
<td>( 1-(1-p_{11})(1-p_{22}) = 2/3 )</td>
</tr>
</tbody>
</table>
Inaccurate Attribute Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Toy</td>
<td>Great</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fair</td>
</tr>
<tr>
<td>Fred</td>
<td>Sales</td>
<td>Good</td>
</tr>
</tbody>
</table>

Inaccurate attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Bonus</th>
<th>E</th>
<th>Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Toy</td>
<td>Great</td>
<td>e_{11}</td>
<td>0.4</td>
</tr>
<tr>
<td>John</td>
<td>Toy</td>
<td>Good</td>
<td>e_{12}</td>
<td>0.5</td>
</tr>
<tr>
<td>John</td>
<td>Toy</td>
<td>Fair</td>
<td>e_{13}</td>
<td>0.1</td>
</tr>
<tr>
<td>Fred</td>
<td>Sales</td>
<td>Good</td>
<td>e_{21}</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Disjoint and/or independent events
Summary on Explicit Tuples

Independent or disjoint/independent tuples
- Possible worlds: subsets
- Probability distribution: restricted
- Closure: no

In KR:
- Bayesian networks: disjoint tuples
- Probabilistic relational models: correlated tuples

[Friedman, Getoor, Koller, Pfeffer: 1999]
Implicit Tuples

“There are other, unknown tuples out there”

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Profession</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Seattle</td>
<td>statistician</td>
</tr>
<tr>
<td>Sue</td>
<td>Boston</td>
<td>musician</td>
</tr>
<tr>
<td>Fred</td>
<td>Boston</td>
<td>Physicist</td>
</tr>
</tbody>
</table>

Covers 10% or Completeness = 10%
Implicit Tuples

Statistics based:

<table>
<thead>
<tr>
<th>Name</th>
<th>Depart</th>
<th>Phone</th>
</tr>
</thead>
</table>

We go with #2: the expected size is $C$
Implicit Possible Tuples

Binomial distribution

\[ E[ \text{Size(Employee)}] = C \]

\[ \forall t. \ Pr(t) = \frac{C}{n_1 n_2 n_3} \]

\[ n_1 = |D_{\text{name}}| \]
\[ n_2 = |D_{\text{dept}}| \]
\[ n_3 = |D_{\text{phone}}| \]
Application 3: Information Leakage

S :- Employee("Mary", -, 5551234)

Pr(S) ≅ C/n_1 n_3

V_1 :- Employee("Mary", "Sales", -)

Pr(S | V_1) ≅ 1/ n_3

Pr(SV_1) ≅ C/n_1 n_2 n_3

Pr(V_1) ≅ C/n_1 n_2

V_2 :- Employee(-, "Sales", 5551234)

Pr(S | V_1 V_2) ≅ 1

Pr(SV_1 V_2) ≅ C/n_1 n_2 n_3

Pr(V_1 V_2) ≅ C/n_1 n_2 n_3

Practical secrecy

Leakage

[Miklau,Dalvi&S:2005]

Pr(name,dept,phone) = C / (n_1 n_2 n_3)
Summary on Implicit Tuples

Given by expected cardinality

• Possible worlds: any
• Probability distribution: binomial

May be used in conjunction with other formalisms

• Entropy maximization

[ Domingos&Richardson:2004, Dalvi&S:2005 ]

Conditional probabilities become important
Summary on Part III: Representation Formalism

• Intensional databases:
  – Complete (in some sense)
  – Impractical, but…
  – …important practical restrictions

• Incomplete formalisms:
  – Explicit tuples
  – Implicit tuples

• We have not discussed query processing yet
Part IV

Foundations
Outline

• Probability of boolean expressions
• Query probability
• Random graphs
Probability of Boolean Expressions

\[ E = X_1X_3 \lor X_1X_4 \lor X_2X_5 \lor X_2X_6 \]

Randomly make each variable true with the following probabilities

\[ \Pr(X_1) = p_1, \ \Pr(X_2) = p_2, \ldots, \Pr(X_6) = p_6 \]

What is \( \Pr(E) \) ???

Answer: re-group cleverly

\[ E = X_1 (X_3 \lor X_4) \lor X_2 (X_5 \lor X_6) \]

\[ \Pr(E) = 1 - \left(1-p_1\left(1-(1-p_3)(1-p_4)\right)\right) \left(1-p_2\left(1-(1-p_5)(1-p_6)\right)\right) \]

Needed for query processing
Now let’s try this:

\[ E = X_1 X_2 \lor X_1 X_3 \lor X_2 X_3 \]

No clever grouping seems possible. Brute force:

\[
\begin{array}{cccccc}
X_1 & X_2 & X_3 & E & \text{Pr} \\
0 & 0 & 0 & 0 & \\
0 & 0 & 1 & 0 & \\
0 & 1 & 0 & 0 & \\
0 & 1 & 1 & 1 & (1-p_1)p_2 p_3 \\
1 & 0 & 0 & 0 & \\
1 & 0 & 1 & 1 & p_1(1-p_2) p_3 \\
1 & 1 & 0 & 1 & p_1 p_2(1-p_3) \\
1 & 1 & 1 & 1 & p_1 p_2 p_3 \\
\end{array}
\]

Seems inefficient in general…
Complexity of Boolean Expression Probability

**Theorem** [Valiant:1979]
For a boolean expression E, computing Pr(E) is \#P-complete

NP = class of problems of the form “is there a witness?” SAT
\#P = class of problems of the form “how many witnesses?” \#SAT

The decision problem for 2CNF is in PTIME
The counting problem for 2CNF is \#P-complete
Summary on Boolean Expression Probability

• #P-complete

• It’s hard even in simple cases: 2DNF

• Can do Monte Carlo simulation (later)
Query Complexity

Data complexity of a query Q:
• Compute $Q(I_p)$, for probabilistic database $I_p$

Simplest scenario only:
• Possible tuples semantics for $Q$
• Independent tuples for $I_p$
Extensional Query Evaluation

Relational ops compute probabilities

Data complexity: PTIME

or: $p_1 + p_2 + \ldots$

SELECT DISTINCT x.City
FROM Person x, Purchase y
WHERE x.Name = y.Cust
and y.Product = ‘Gadget’

[Dalvi&S:2004]

Wrong!

Correct

Depends on plan!!!
Query Complexity

Sometimes $\not\exists$ correct extensional plan

$$Q_{\text{bad}} :- R(x), S(x,y), T(y)$$

Data complexity is #P complete

Theorem The following are equivalent

- $Q$ has PTIME data complexity
- $Q$ admits an extensional plan (and one finds it in PTIME)
- $Q$ does not have $Q_{\text{bad}}$ as a subquery

[Dalvi&S:2004]
Summary on Query Complexity

Extensional query evaluation:
• Very popular
• However, result depends on query plan!

General query complexity
• \#P complete (not surprising, given \#SAT)
• Already \#P hard for very simple query (Q_{bad})

Probabilistic database have high query complexity
Random Graphs

Relation:

Domain:

$G^p = \text{tuple-independent}$

Graph $G$:

Random graph $G^p$

Boolean query $Q$

What is $\lim_{n \to \infty} Q(G^p)$
**Fagin’s 0/1 Law**

Let the tuple probability be \( p = 1/2 \)

**Theorem** [Fagin:1976,Glebskii et al.1969]

For every sentence \( Q \) in First Order Logic,

\[
\lim_{n \to \infty} Q(G^p) \text{ exists and is either 0 or 1}
\]

Holds almost surely:

\[
\lim = 1
\]

Does not hold a.s. \( \lim = 0 \)

**Examples**

<table>
<thead>
<tr>
<th>Holds almost surely:</th>
<th>( \lim = 1 )</th>
<th>Does not hold a.s.</th>
<th>( \lim = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \forall x. \exists y. G(x,y) )</td>
<td>( \exists x. \forall y. G(x,y) )</td>
<td>( \forall x. \forall y. G(x,y) )</td>
<td></td>
</tr>
</tbody>
</table>
Erdos and Reny’s Random Graphs

Now let $p = p(n)$ be a function of $n$

Theorem [Erdos & Reny: 1959]
For any monotone $Q$, $\exists$ a threshold function $t(n)$ s.t.:

- if $p(n) \ll t(n)$ then $\lim_{n \to \infty} Q(G_p) = 0$
- if $p(n) \gg t(n)$ then $\lim_{n \to \infty} Q(G_p) = 1$
The Evoluation of Random Graphs

The tuple probability \( p(n) \) "grows" from 0 to 1. How does the random graph evolve?

Remark: \( C(n) = E[ \text{Size}(G) ] \approx n^2 p(n) \)

The expected size \( C(n) \) "grows" from 0 to \( n^2 \). How does the random graph evolve?
The Void

\[ p(n) \ll \frac{1}{n^2} \]

\[ C(n) \ll 1 \]

<table>
<thead>
<tr>
<th>Contains almost surely</th>
<th>Does not contain almost surely</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nothing)</td>
<td></td>
</tr>
</tbody>
</table>

The graph is empty

0/1 Law holds
On the k’th Day

\[
\frac{1}{n^{1+1/(k-1)}} \ll p(n) \ll \frac{1}{n^{1+1/k}}
\]

\[
n^{1-1/(k-1)} \ll C(n) \ll n^{1-1/k}
\]

<table>
<thead>
<tr>
<th>Contains almost surely</th>
<th>Does not contain almost surely</th>
</tr>
</thead>
<tbody>
<tr>
<td>trees with ≤ k edges</td>
<td>trees &gt; k edges</td>
</tr>
<tr>
<td>[Diagram of trees]</td>
<td>[Diagram of trees with cycles]</td>
</tr>
</tbody>
</table>

The graph is disconnected

0/1 Law holds

[Spencer:2001]
On Day $\omega$

\[
\frac{1}{n^{1+\epsilon}} \ll p(n) \ll \frac{1}{n}, \quad \forall \epsilon > 0
\]

\[
n^{1-\epsilon} \ll C(n) \ll n, \quad \forall \epsilon > 0
\]

<table>
<thead>
<tr>
<th>Contains almost surely</th>
<th>Does not contain almost surely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any Tree</td>
<td>cycles</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The graph is disconnected  0/1 Law holds
Past the Double Jump (1/n)

\[ \frac{1}{n} \ll p(n) \ll \frac{\ln(n)}{n} \quad \text{and} \quad n \ll C(n) \ll n \ln(n) \]

<table>
<thead>
<tr>
<th>Contains almost surely</th>
<th>Does not contain almost surely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any Tree</td>
<td>Any subgraph with k nodes and ( \geq k+1 ) edges</td>
</tr>
<tr>
<td>Any Cycle</td>
<td></td>
</tr>
</tbody>
</table>

The graph is disconnected  0/1 Law holds

[Spencer:2001]
# Past Connectivity

\[
\ln(n)/n \ll p(n) \ll 1/n^{1-\varepsilon}, \forall \varepsilon
\]

\[
n \ln(n) \ll C(n) \ll n^{1+\varepsilon}, \forall \varepsilon
\]

## Contains almost surely

- Every node has degree \( \geq k \), for every \( k \geq 0 \)

## Does not contain almost surely

- Any subgraph with \( k \) nodes and \( \geq k+1 \) edges

**Strange logic of random graphs !!**

---

The graph is connected !

0/1 Law holds
Big Graphs

\[ p(n) = \frac{1}{n^\alpha}, \quad \alpha \in (0,1) \]

\[ C(n) = n^{2-\alpha}, \quad \alpha \in (0,1) \]

\( \alpha \) is irrational \( \Rightarrow \) 0/1 Law holds

\( \alpha \) is rational \( \Rightarrow \) 0/1 Law does not hold

Fagin’s framework: \( \alpha = 0 \)

\[ p(n) = O(1) \]

\[ C(n) = O(n^2) \]
Summary on Random Graphs

• Very rich field
  – Over 700 references in [Bollobas:2001]

• Fascinating theory
  – Evening reading: the evolution of random graphs (e.g. from [Spencer:2001])
Summary on Random Graphs

• Fagin’s 0/1 Law: impractical probabilistic model

• More recent 0/1 laws for \( p = 1/n^\alpha \)
  [Spencer&Shelah, Lynch]

• In practice: need precise formulas for \( \Pr(Q(I^p)) \)
  – Preliminary work [Dalvi,Miklau&S:04,Dalvi&S:05]
Part V

Algorithms,
Implementation Techniques
Query Processing on a Probabilistic Database

1. Simulation
2. Extensional joins
3. Indexes
1. Monte Carlo Simulation

Naïve:

\[ E = X_1X_2 \lor X_1X_3 \lor X_2X_3 \]

\[
\text{Cnt} \leftarrow 0 \\
\text{repeat} \ N \ \text{times} \\
\quad \text{randomly choose } X_1, X_2, X_3 \in \{0,1\}
\]

\[
\text{if } E(X_1, X_2, X_3) = 1 \\
\quad \text{then } \text{Cnt} = \text{Cnt} + 1
\]

\[ P = \frac{\text{Cnt}}{N} \]

\text{return } P /* \approx \text{Pr}(E) */

**Theorem.** If \( N \geq \left(1/ \text{Pr}(E)\right) \times (4\ln(2/\delta)/\epsilon^2) \) then:

\[ \text{Pr}[ | P/\text{Pr}(E) - 1 | > \epsilon ] < \delta \]

Works for any \( E \) Not in PTIME

May be very big

0/1-estimator theorem
Monte Carlo Simulation

Improved:

\[ E = C_1 \lor C_2 \lor \ldots \lor C_m \]

\[
\text{Cnt} \leftarrow 0; \quad S \leftarrow \Pr(C_1) + \ldots + \Pr(C_m);
\]

\textbf{repeat} \ N \textbf{times}

- randomly choose \( i \in \{1, 2, \ldots, m\} \), with prob. \( \Pr(C_i) / S \)
- randomly choose \( X_1, \ldots, X_n \in \{0, 1\} \) s.t. \( C_i = 1 \)
- \textbf{if} \( C_1 = 0 \) and \( C_2 = 0 \) and \( \ldots \) and \( C_{i-1} = 0 \)
  \textbf{then} \ \text{Cnt} = \text{Cnt} + 1

\[
P = \frac{\text{Cnt}}{N} \times \frac{1}{\varepsilon}
\]

\textbf{return} \ \frac{P}{\varepsilon} /* \sim \Pr(E) */

\textbf{Theorem.} If \( N \geq \left( \frac{1}{m} \right) \times \left( \frac{4\ln(2/\delta)}{\varepsilon^2} \right) \) then:

\[
\Pr[ \mid P/\Pr(E) - 1 \mid > \varepsilon ] < \delta
\]

Only for \( E \) in DNF

In PTIME
Summary on Monte Carlo

Some form of simulation is needed in probabilistic databases, to cope with the #P-hardness bottleneck

- Naïve MC: works well when Prob is big
- Improved MC: needed when Prob is small
2. The Threshold Algorithm

Problem:

SELECT *
FROM Rp, Sp, Tp

Have subplans for Rp, Sp, Tp returning tuples sorted by their probabilities x, y, z

Score combination:
f(x, y, z) = xyz

How do we compute the top-k matching records?
“No Random Access” (NRA)

Total score:

<table>
<thead>
<tr>
<th>Rp</th>
<th>Sp</th>
<th>Tp</th>
</tr>
</thead>
<tbody>
<tr>
<td>H32</td>
<td>x₁</td>
<td>H44</td>
</tr>
<tr>
<td>H5555</td>
<td>x₂</td>
<td>H007</td>
</tr>
<tr>
<td>H007</td>
<td>x₃</td>
<td>H999</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

1 ≥ x₁ ≥ x₂ ≥ ...

1 ≥ y₁ ≥ y₂ ≥ ...

1 ≥ z₁ ≥ z₂ ≥ ...

H???: f(?, ?, ?)

H007: f(x₃,?, z₂)

H5555: f(x₂,y₁,z₃)

H999: f(?, y₃, ?)

H32: f(x₁,?, ?)

H5555: f(x₂,y₁,z₃)

0 ≤ ? ≤ y₃
Termination condition:

\[ H?:?:?: f(?, ?, ?) \]

Threshold score

k objects:
Guaranteed to be top-k

The algorithm is “instance optimal”
strongest form of optimality
Summary on the Threshold Algorithm

• Simple, intuitive, powerful
• There are several variations: see paper
• Extensions:
  – Use probabilistic methods to estimate the bounds more aggressively
    [Theobald, Weikum & Schenkel: 2004]
  – Distributed environment
    [Michel, Triantafillou & Weikum: 2005]
Approximate String Joins

Problem:

```
SELECT *
FROM R, S
WHERE R.A ~ S.B
```

Simplification for this tutorial:
A ~ B means “A, B have at least k q-grams in common”
Definition of q-grams

String: John_Smith

Set of 3-grams:

##J  #Jo  Joh  ohn  hn_  n_S  _Sm  Smi  mit  ith  th#  h##
Naïve solution, using UDF (user defined function)

```
SELECT *
FROM R, S
WHERE common_grams(R.A, S.B) ≥ k
```
A q-gram index:

| Key  | A        | ...
|------|----------|------------------
| ...  |          |                  |
| k743 | John_Smith | ... |
| ...  |          |                  |

[Gravano et al.:2001]

RAQ

<table>
<thead>
<tr>
<th>Key</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>k743</td>
<td>##J</td>
</tr>
<tr>
<td>k743</td>
<td>#Jo</td>
</tr>
<tr>
<td>k743</td>
<td>Joh</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Solution using the Q-gram Index

\[
\text{SELECT} \, * \\
\text{FROM} \, R, \, S \\
\text{WHERE} \, R.A \sim S.B
\]

\[
\text{SELECT} \, R.*, \, S.* \\
\text{FROM} \, R, \, RAQ, \, S, \, SBQ \\
\text{WHERE} \, R.Key = RAQ.Key \, \text{and} \, S.Key=SBQ.Key \\
\text{and} \, RAQ.G = RBQ.G \\
\text{GROUP BY} \, RAQ.Key, \, RBQ.Key \\
\text{HAVING} \, \text{count}(*) \geq k
\]

[Gravano et al.:2001]
Summary on Part V: Algorithms

A wide range of disparate techniques

• Monte Carlo Simulations (also: MCMC)
• Optimal aggregation algorithms (TA)
• Efficient engineering techniques

Needed: unified framework for efficient query evaluation in probabilistic databases
Conclusions and Challenges Ahead
Conclusions

Imprecisions in data:

• A wide variety of types have specialized management solutions

• Probabilistic databases promise uniform framework, but require full complexity
Conclusions

Probabilistic databases

• Possible worlds semantics
  – Simple
  – Every query has well defined semantics

• Need: expressive representation formalism

• Need: efficient query processing techniques
Challenge 1:
Specification Frameworks

The Goal:
• Design framework that is usable, expressive, efficient

The Challenge
• Tradeoff between expressibility and tractability
Challenge 1: Specification Frameworks

Features to have:

- **Support probabilistic statements:**
  - Simple: \((\text{Fred}, \text{Seattle}, \text{Gizmo}) \in \text{Purchase}\) has probability 60%
  - Complex: “Fred and Sue live in the same city” has probability 80%

- **Support tuple correlations**
  - “\(t_1\) and \(t_2\) are correlated positively 30%”

- **Statistics statements:**
  - There are about 2000 tuples in Purchase
  - There are about 100 distinct Cities
  - Every customer buys about 4 products

[Domingos&Richardson:04,Sarma,Benjelloun,Halevy,Widom:2005]
Challenge 2: Query Evaluation

Complexity

• Old: $= f(\text{query-language})$
• New: $= f(\text{query-language}, \text{specification-language})$

Exact algorithm: $\#P$-complete in simple cases

Challenge: characterize the complexity of approximation algorithms
Challenge 2: Query Evaluation

Implementations:
• Disparate techniques require unified framework
• Simulation:
  – Client side or server side?
  – How to schedule simulation steps?
  – How to push simulation steps in relational operators?
  – How to compute subplans extensionally, when possible?
• Top-k pruning:
  – How can we “push thresholds” down the query plan?
Challenge 3: Mapping Imprecisions to Probabilities

- One needs to put a number between 0 and 1 to an uncertain piece of data
  - This is highly nontrivial!
  - But consider the alternative: ad-hoc management of imprecisions at all stages

- What is a principled approach to do this?
- How do we evaluate such mappings?
The End

Questions ?